I still need to show you that this construction actually does what it is supposed to do.

**E.g.** Some string accepted by the PDA

\[ w = aabb \]

Instantaneous descr.

\[
(q_0, aabb, z) \quad \rightarrow \quad (q_1, A, z) \quad \rightarrow \quad (q_1, bb, AAz) \quad \rightarrow \quad (q_2, b, AZ) \quad \rightarrow \quad (q_2, \lambda, Z) \quad \rightarrow \quad (q_f, \lambda, \lambda)
\]

This "proves" (to do it properly I should do it abstractly for every sequence of instantaneous descriptions) that every string accepted by the PDA is generated by the grammar.
I should now show you why every derivation in the grammar corresponds
to a sequence of instant.
descr. for the PDA.

1. Make sure we use a leftmost derivation

2. The transition we use at any given point is just the transition corresponding to the production (we don't have a choice - there is only one)
Deterministic PDAs

A PDA is deterministic if

1. We do not have two different transitions
\[(q, l, s) \rightarrow (q', t')\]
\[(q, l, s) \rightarrow (q'', t'')\]
\quad (where \(q' \neq q''\) or \(t' \neq t''\))

the state we’re at,

Intuitively, given the first letter remaining on the string and the top of the stack, we have at most one choice.

and - (we want \(\lambda\)-transitions so we can do stack stuff w/o eating string, but we don’t want a choice btw a \(\lambda\)-transition and a string eating one.

2. We do not have transitions
\[(q, \lambda, s) \rightarrow (q', t')\]
\[(q, l, s) \rightarrow (q'', t'')\] (where \(l \neq \lambda\))